

Python: module `cdms.database`

cdms.database

[index](#)

CDMS database objects

Modules

<u>cdms.cdmsobj</u>	<u>cdms.internattr</u>	<u>string</u>
<u>cdms.cdurlparse</u>	<u>os</u>	<u>sys</u>
<u>copy</u>	<u>re</u>	<u>types</u>

Classes

[cdms.cdmsobj.CdmsObj\(cdms.internattr.InternalAttributesClass\)](#)

[AbstractDatabase](#)

[LDAPDatabase](#)

[AbstractResultEntry](#)

[LDAPResultEntry](#)

[AbstractSearchResult](#)

[LDAPSearchResult](#)

class [AbstractDatabase\(cdms.cdmsobj.CdmsObj\)](#)

[AbstractDatabase](#) defines the common database interface. Concrete database classes derived from this class.

Method resolution order:

[AbstractDatabase](#)

[cdms.cdmsobj.CdmsObj](#)

[cdms.internattr.InternalAttributesClass](#)

[PropertiedClasses.Properties.PropertiedClass](#)

Methods defined here:

[__init__](#)(self, uri, path)

[__repr__](#)(self)

[cachedmml](#)(self, name, cdml)

[close](#)(self)

disableCache(self)

enableCache(self)

getDataset(self, name)

getObjFromDataset(self, name)

openDataset(self, dsetid, mode='r')

searchFilter(self, filter, classtag=None, relbase=None, scope=2, attnames=[])

useRequestManager(self, lcBaseDN, useReplica=1, userid='anonymous')

usingRequestManager(self)

Methods inherited from cdms.cdmsobj.CdmsObj:

dump(self, path=None, format=1)

`dump`(self, path=None, format=1)

Dump an XML representation of this object to a file.

'path' is the result file name, None for standard output.

'format'==1 iff the file is formatted with newlines for readability.

matchPattern(self, pattern, attribute, tag)

Match a pattern in a string-valued attribute. If attribute is None, search all string attributes.

If tag is not None, it must match the attribute.

matchone(self, pattern, atname)

Return true iff the attribute with name atname is a string-valued attribute which matches the compiled regular expression pattern.

If atname is None and pattern matches at least one string-valued attribute. Return false if the attribute is not found or is not a string.

searchPattern(self, pattern, attribute, tag)

Search for a pattern in a string-valued attribute. If attribute is None, search all string attributes.

If tag is not None, it must match the attribute.

searchPredicate(self, predicate, tag)

Apply a truth-valued predicate. Return a list containing a string-valued attribute which matches the predicate.

If the predicate is true and either tag is None or matches the attribute.

If the predicate returns false, return an empty list

searchone(self, pattern, atname)

Return true iff the attribute with name atname is a string-valued attribute which contains the compiled regular expression pattern.

if atname is None and pattern matches at least one string-valued attribute. Return false if the attribute is not found or is not a string.

Methods inherited from cdms.internattr.InternalAttributesClass:

is_internal_attribute(self, name)

`is_internal_attribute(name)` is true if name is internal.

replace_external_attributes(self, newAttributes)

`replace_external_attributes(newAttributes)`

Replace the external attributes with dictionary newAttributes

Methods inherited from PropertiedClasses.Properties.PropertiedClass:

`__delattr__`(self, name)

`__getattr__`(self, name)

`__setattr__`(self, name, value)

get_property_d(self, name)

Return the 'del' property handler for name that self uses.
Returns None if no handler.

get_property_g(self, name)

Return the 'get' property handler for name that self uses.
Returns None if no handler.

get_property_s(self, name)

Return the 'set' property handler for name that self uses.
Returns None if no handler.

set_property(self, name, actg=None, acts=None, actd=None, nowrite=None, nodelete=None)

Set attribute handlers for name to methods actg, acts, actd
None means no change for that action.

nowrite = 1 prevents setting this attribute.

nowrite defaults to 0.

nodelete = 1 prevents deleting this attribute.

nodelete defaults to 1 unless actd given.

if nowrite and nodelete is None: nodelete = 1

class ***AbstractResultEntry***

Methods defined here:

`__init__`(self, db)

getObject(self)

Method:

`getObject()`

Description:

Get the CDMS object associated with this entry.

Returns:

Instance of a CDMS object.

class ***AbstractSearchResult***

Methods defined here:

__getitem__(self, key)

__len__(self)

searchPredicate(self, predicate, tag=None)

class ***LDAPDatabase***(AbstractDatabase)

Database implemented via LDAP (Lightweight Directory Access Protocol)

Method resolution order:

LDAPDatabase

AbstractDatabase

cdms.cdmsobj.CdmsObj

cdms.internattr.InternalAttributesClass

PropertiedClasses.Properties.PropertiedClass

Methods defined here:

__del__(self)

__init__(self, uri, db)

cachedmml(self, name, cdml, datapath)

close(self)

Method:

close ()

Description:

Close a database connection.

Returns:

None

getDataset(self, dn)

getObjFromDataset(self, dn)

listDatasets(self)

Return a list of the dataset IDs in this database.

normalizedn(self, dn)

open = openDataset(self, dsetid, mode='r')

openDataset(self, dsetid, mode='r')

Method:

```
openDataset(dsetid, mode='r')
```

Description:

Open a dataset.

Arguments:

```
dsetid: string dataset identifier
mode: open mode ('r' - read-only, 'r+' - read-write, 'w' -
```

Returns:

Dataset instance.

Example:

```
dset = db.openDataset('ncep_reanalysis_mo')
```

searchFilter(self, filter=None, tag=None, relbase=None, scope=2, attnames=None, timeout=None)

Method:

```
searchFilter(filter=None, tag=None, relbase=None, scope=Sub
```

Description:

Search a CDMS database.

Arguments:

```
filter: string search filter
Simple filters have the form "tag = value". Simple filter
logical operators '&', '|', '!' in prefix notation. For e
the filter '(&(objectclass=variable)(id=cli))' finds all
```

More formally:

```
filter      ::= "(" filtercomp ")"
filtercomp ::= "&" filterlist | # and
              "|" filterlist | # or
              "!" filterlist | # not
              simple
```

```

filterlist ::= filter | filter filterlist
simple      ::= tag op value
op         ::= "=" |          # equality
           ::= "~=" |        # approximate equality
           ::= "<=" |         # lexicographically less than
           ::= ">=" |         # lexicographically greater than
value      ::= string, may include '*' as a wild card

```

```

tag: string class tag ("dataset" | "variable" | "database" | ...)
Restricts the search to a class of objects
relbase: string search base, relative to the database path
scope: search scope (Subtree | Onelevel | Base). Subtree searches
       the subtree rooted at the base object and its immediate descendants.
       Onelevel searches the base object and its immediate descendants.
       Default is Subtree.
attnames: list of attribute names. Restricts the attributes returned.
timeout: integer number of seconds before timeout.

```

Returns:

```

SearchResult instance. Entries can be accessed sequentially.
entry.name returns the name of the entry, entry.attributes is a dictionary of the
entry.getObjct() returns the CDMS object associated with the entry.

```

```

for entry in result:
    print entry.name, entry.attributes["id"]

```

Entries can be refined with [searchPredicate\(\)](#).

Example:

(1) Find all variables named "cli":

```
result = db.searchFilter(filter="id=cli",tag="variable")
```

(2) Find all objects in dataset "ncep_reanalysis_mo":

```
result = db.searchFilter(relbase="dataset=ncep_reanalysis_mo")
```

setExternalDict(self, ldapattrs)

```

# Set the database attributes from an LDAP search result.
# ldapattrs is a dictionary, keyed on attribute name.
# Values are lists of attribute values.

```

Methods inherited from [AbstractDatabase](#):

__repr__(self)

disableCache(self)

enableCache(self)

useRequestManager(self, lcBaseDN, useReplica=1, userid='anonymous')

usingRequestManager(self)

Methods inherited from cdms.cdmsobj.CdmsObj:

dump(self, path=None, format=1)

dump(self, path=None, format=1)

Dump an XML representation of this object to a file.

'path' is the result file name, None for standard output.

'format'==1 iff the file is formatted with newlines for readability.

matchPattern(self, pattern, attribute, tag)

Match a pattern in a string-valued attribute. If attribute is None, search all string attributes. If tag is not None, it must match the tag.

search all string attributes. If tag is not None, it must match the tag.

matchone(self, pattern, attname)

Return true iff the attribute with name attname is a string-valued attribute which matches the compiled regular expression pattern.

attribute which matches the compiled regular expression pattern.

if attname is None and pattern matches at least one string-valued attribute.

attribute. Return false if the attribute is not found or is not a string.

searchPattern(self, pattern, attribute, tag)

Search for a pattern in a string-valued attribute. If attribute is None, search all string attributes. If tag is not None, it must match the tag.

search all string attributes. If tag is not None, it must match the tag.

searchPredicate(self, predicate, tag)

Apply a truth-valued predicate. Return a list containing all string-valued attributes which match the predicate.

if the predicate is true and either tag is None or matches the tag.

If the predicate returns false, return an empty list

searchone(self, pattern, attname)

Return true iff the attribute with name attname is a string-valued attribute which contains the compiled regular expression pattern.

attribute which contains the compiled regular expression pattern.

if attname is None and pattern matches at least one string-valued attribute.

attribute. Return false if the attribute is not found or is not a string.

a string.

Methods inherited from cdms.internattr.InternalAttributesClass:

is_internal_attribute(self, name)

is internal attribute(name) is true if name is internal.

replace_external_attributes(self, newAttributes)

replace external attributes(newAttributes)

Replace the external attributes with dictionary newAttributes

Methods inherited from PropertiedClasses.Properties.PropertiedClass:

delattr(self, name)

getattr(self, name)

`__setattr__(self, name, value)`

`get_property_d(self, name)`

Return the 'del' property handler for name that self uses.
Returns None if no handler.

`get_property_g(self, name)`

Return the 'get' property handler for name that self uses.
Returns None if no handler.

`get_property_s(self, name)`

Return the 'set' property handler for name that self uses.
Returns None if no handler.

`set_property(self, name, actg=None, acts=None, actd=None, nowrite=None, nodelete=None)`

Set attribute handlers for name to methods actg, acts, actd
None means no change for that action.
nowrite = 1 prevents setting this attribute.
nowrite defaults to 0.
nodelete = 1 prevents deleting this attribute.
nodelete defaults to 1 unless actd given.
if nowrite and nodelete is None: nodelete = 1

class **`LDAPResultEntry`**(AbstractResultEntry)

Methods defined here:

`__init__(self, db, dn, attributes)`

Methods inherited from AbstractResultEntry:

`getObject(self)`

Method:

`getObject()`

Description:

Get the CDMS object associated with this entry.

Returns:

Instance of a CDMS object.

class **`LDAPSearchResult`**(AbstractSearchResult)

Methods defined here:

`__getitem__(self, key)`

`__init__(self, db, LDAPresult)`

`__len__(self)`

`searchPredicate(self, predicate, tag=None)`

Method:

`searchPredicate(predicate, tag=None)`

Description:

Refine a search result, with a predicate search.

Arguments:

`predicate`: Function name or lambda function. The function takes an object and returns true (1) if the object satisfies the predicate.
`tag`: Restrict the search to objects in one class.

Returns:

SearchResult instance. Entries can be accessed sequentially. `entry.name` is the name of the entry, `entry.attributes` is a dictionary of the attributes. `entry.getObject()` returns the CDMS object associated with the entry.

```
for entry in result:
    print entry.name, entry.attributes["id"]
```

Entries can be refined with `searchPredicate()`.

Example:

(1) Find all variables on a 73x96 grid

```
newresult = result.searchPredicate(lambda obj: obj.getGrid()
```

Functions

`connect(uri=None, user="", password="")`

Method:

`connect(uri=None, user="", password="")`

Description:

Open a CDMS database connection.

Arguments:

`uri`: Universal Resource Identifier. If unspecified, defaults to

```
user: user id
password: password
```

Returns:

Database instance.

Example:

```
db = cdms.connect("ldap://dbhost.llnl.gov/database=CDMS,ou=PCMDI")
```

loadString(text, uri, parent=None, datapath=None)

Create a dataset from a text string. <text> is the string in CDML

<uri> is the URL of the dataset in a catalog or file.

<parent> is the containing database object, if any.

<datapath> is the location of data files relative to the parent da

Data

AuthenticationError = 'Error authenticating to database'

Base = 0

CannotOpenDataset = 'Cannot open dataset'

ConnectError = 'Error connecting to database'

DatabaseNotFound = 'Database not found'

InvalidEntryName = 'Invalid entry name'

MethodNotImplemented = 'Method not yet implemented'

Onelevel = 1

PermissionError = 'No permission to access'

SchemeNotSupported = 'Scheme not supported'

Subtree = 2